Army Research Laboratory

U.S. Army Research Laboratory (ARL) *XPairIt* Simulator for Peptide Docking and Analysis

by Michael S. Sellers and Margaret M. Hurley

ARL-TR-6999 July 2014

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5069

ARL-TR-6999 **July 2014**

U.S. Army Research Laboratory (ARL) *XPairIt* Simulator for Peptide Docking and Analysis

Michael S. Sellers and Margaret M. Hurley
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* July 2014	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** October 2010–October 2012
4. TITLE AND SUBTITLE U.S. Army Research Laboratory (ARL) *XPairIt* Simulator for Peptide Docking and Analysis		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Michael S. Sellers and Margaret M. Hurley		**5d. PROJECT NUMBER** BRCALL08-PER3
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-WML-B Aberdeen Proving Ground, MD 21005-5069		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-6999
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Identifying the natural orientation of a peptide bound to a protein via computational means presents unique obstacles not often encountered in the more common searches of ligand-protein or protein-protein complexes. Along with a brief review of the current state of methods in these simulations, we introduce software to tackle many of the challenges that still exist. These hurdles are addressed with the *XPairIt API*, a docking protocol which unites powerful open source simulation packages and enhances the representation of energetics and flexibility within a simulation. Ability of the protocol is demonstrated through a global docking case study, and we show improvement in binding energy with the combined use of molecular dynamics and Monte Carlo docking. Comparison of methodology and final root mean squared displacement (RMSD) of our bound structures to previous work is made, and we highlight improvements in peptide and protein flexibility.

15. SUBJECT TERMS
peptide docking, docking methodology, software design

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Michael S. Sellers
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** (*Include area code*)
Unclassified	Unclassified	Unclassified	UU	34	410-306-0728

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to acknowledge and thank their experimental collaborators, Dimitra Stratis-Cullum, Bryn Adams, Joshua Kogot, Deborah Sarkes, Irene Val-Addo, and Paul Pellegrino, of the U.S. Army Research Laboratory (ARL), Sensors and Electronic Devices Directorate, and Rebecca Brown, Candice Warner, and James Carney of the U.S. Army Edgewood Chemical and Biological Center.

The authors gratefully acknowledge use of high-performance computing (HPC) resources administered by the Department of Defense High-Performance Computing Modernization program.

This project receives support from the Defense Threat Reduction Agency Joint Science and Technology Office for Chemical and Biological Defense (Grant no. BRCALL08-Per3-P-2-0028). Research is also supported in part by appointments to ARL Postdoctoral Fellowship Program administered by the Oak Ridge Associated Universities through a contract with ARL.

Images of protein structures were created using *Visual Molecular Dynamics* (*VMD*), and plots were made with an *XPairIt* interface to the Matplotlib Python library (*1*, *2*).

1. Introduction and Background

As computational biology continues to play a larger role in applied research, collaborations between simulation and experiment can benefit from software that provides access to various styles of methods, enabling customization and rapid application to a wide range of biological systems. A key work by Elcock, et al. sets the stage in computational biology for this style of adaptable research, early on in the last decade (*3*). In their investigation of the driving forces in protein-protein interactions, the authors draw upon several computational methods to characterize interaction energy and orientation, and their effects on macroscopic quantities like the second-order virial coefficient. Years later, researchers have at their disposal many software options for tackling the handful of diverse systems involving biomolecular interactions. However, features that make software in computational biology accessible, usable, and powerful remain characteristic of a select few (*4–9*). Therefore, the challenge presented to this decade's researcher is how to efficiently use these software packages in concert to study aspects of a biological system.

We set our focus on open-source simulation packages, and unite these powerful, sometimes disparate modes of simulation with a simple, extensible, and object-oriented Python suite of code called *XPairIt*. Joining such programs provides enhanced simulation functionality through a concerted use of each program's multiple length scales, interatomic potentials, and simulation methods. This is accomplished with the parallelized core of *XPairIt*, which additionally contains many data management and organization options, analysis tools, and custom simulation methodology. Two open-source simulation APIs offer options similar to *XPairIt*. The first is the *Molecular Modeling Toolkit* (MMTK) (*10*). MMTK, initially developed about a decade ago, is written in Python using object-oriented design, and is a more stand-alone piece of software when compared to *XPairIt*. It incorporates internal molecular dynamics and Monte Carlo algorithms, several interatomic potentials, and normal mode analysis into a single software package for the simulation of protein systems. The second is *SimTK's OpenMM*, which is another general simulation toolkit using graphics processing unit (GPU) hardware acceleration (*11*). These toolkits differ from *XPairIt* in that much of the simulation in *MMTK* and *OpenMM* is done within the core of the toolkit, where in *XPairIt* the simulation engines are typically external software packages. In this application of the *XPairIt API*, we create a freeware, extensible, high-performance computing (HPC)-ready, multi-scale biomolecular simulator built to include several industry proven external software packages to tackle many of the current challenges in modeling and analyzing peptide-protein complexes.

The main simulation field specializing in the investigation of peptide-protein interactions is known as "docking." That is, the efficient simulation of the interaction of two biomolecules to determine their natural and preferred orientation when in contact with one another. Some of the multi-method functionality for molecular docking offered by *XPairIt* is currently available in a few commercial software suites, such as *Molecular Operating Environment* (*MOE*) or *Shrodinger* through its *PIPER* and *JAGUAR* modules (*8*, *9*). While these software packages are largely successful in modeling many types of biomolecular interactions, there are several shortcomings which become apparent when applied to the docking of certain molecules, such as peptides (*12*, *13*).

Recent software innovations have attempted to address the challenges of incorporating peptide flexibility in docking simulations and adequately ranking ("scoring") the thousands of generated structures to determine the likely binding location. Many use the *Rosetta Modeling Suite*, which is a software package for protein structure prediction, protein-protein and small molecule docking, and protein design (*7*, *14*). This was used to study protein-protein docking using only fixed protein displacement and side-chain rotamers, but the authors highlight possible improvement by allowing for a flexible backbone (*15*, *16*). Rosetta was also used to study peptides where a starting peptide-protein crystal structure was known, and small Monte Carlo (MC) simulations of the peptide backbone were incorporated. The resulting docked structures exhibited more hydrogen bonds and better van der Waals contact (*17*). Other modes of *Rosetta* have also been used to improve docking. For example, Sammond, et al. use a combination of MC backbone moves and the design feature of *Rosetta* to optimize peptide structure and sequence when bound to a G-protein α subunit (*18*). Additionally, work by Raveh, et al. incorporates a more robust design scheme of *Rosetta* to optimize peptide structure with multi-residue fragment building (*19*, *20*). Here, the peptide sequence is fixed and *Rosetta* is used to sample various groups of backbone dihedral angles—effectively using MC tests of α, β, or coil structures on 3 to 4 residue-length clusters of the peptide. They also incorporate a scoring scheme using multiple rounds of rigid body docking, while actively changing the weights of the attractive and repulsive terms of the *Rosetta score12* score function. A validation of this method on about 20 peptide-protein crystal structures shows promising results.

Other simulation methods have been used to improve flexibility and scoring/ranking as well. Use of molecular dynamics software in combination with a docking program has emerged recently as a technique complementary to all-Monte Carlo style docking. Variations of this method are outlined in recent review articles (*21*, *22*). One of the early uses of molecular dynamics in docking was shown in Lin *et al.,* where flexibility of the protein receptor is captured through long molecular dynamics simulations and ligands are then docked to the ensemble of receptor configurations (*23*). Later on, a multi-software method was used by Okimoto et al. to combine molecular dynamics and solvation energy computation into docking small ligands (*24*). The authors used *GOLD* to first dock the ligand, and then minimized the outputted structure using a molecular mechanics forcefield. The minimized structure was then simulated with *AMBER 8.0 (ff03)* using molecular dynamics on add-in computational hardware called *MDGRAPE-3*. Finally, the structure was scored with *MM/PB-SA* to find the ΔG of binding by computing the conformational energy of the ligand, nonbonded van der Waals and electrostatic interactions, the solvation free energy, and the nonpolar solvation free energy. For a sample of 1000 ligands, including 30% active ligands, the authors report improved results for three target proteins, with the exception of *CDK2*.

In a another approach, Antes presented *DynaDock* in 2009, a joint dynamics and docking software package using customized interatomic potentials via the *OPMD* approach (*25*). *OPMD* was used to sample multiple energy minima around a crystal structure configuration with dynamics, and did not rely on annealing methods. Antes also incorporated a custom scoring function, combining the conformational energy of the peptide, with nonbonded van der Waals and Coulombic interactions with the protein. When compared to *AutoDock*, the method performed better for peptides where the number of rotatable bonds is larger than 15 (*26*). Finally, Dagliyan et al. used several molecular dynamics methods, without formal docking software, to generate bound configurations of peptide protein complexes (*27*). Without knowledge of the bound crystal structure, replica exchange dynamics coupled with discrete molecular dynamics integration is used to simulate various peptides at random points around the protein surface. The authors then compute the binding energy for these structures with the *MedusaDock* scoring function. Results suggest that electrostatics play an important role in the formation of an "encounter complex" prior to forming a bound conformation.
In addition to molecular dynamics and mechanics, use of different length and timescale methods can improve flexibility and scoring/ranking as well. Nowosielski et al. connected quantum level energetics to molecular dynamics and docking methods to study ligand binding in *pantothenate synthetase* and successfully simulated the open-to-closed transition of the enzyme for ligand binding (*28*). To achieve longer timescale simulations, the steered-molecular dynamics (SMD) method was used on various ligand-enzyme complexes by Whalen et al. (*29*) SMD simulations captured ligand binding energies by inducing a transition from ligand-bound and enzyme-closed to ligand-unbound and enzyme-open.

Although these examples incorporate dynamics slightly differently, they all show good improvement on statically docked structures when applying dynamics in lieu of Monte Carlo moves. It is also important to note that only one of the previous docking methods outlined start without a crystal structure, and many do not attempt docking peptides with unknown binding locations. Even so, these works do highlight the benefits of using dynamics, and also exhibit a need for a generalized, usable code to connect different styles of simulation software. In recent works by Seeliger et al. and Lill et al., the authors connect the popular PyMol visualization and analysis program to docking and molecular dynamics software (*30–32*). With this software, researchers can conduct simulation and analysis of ligand binding and design within PyMol using a graphical interface. Professional 3D modeling applications, such as *Blender* and *Autodesk 3DS Max*[*] are unique areas for connection to molecular docking. Users can add a plug-in from the Olsen Laboratory at the *Scripps Research Institute* called *ePMV* to dock biological structures and compute energies (*33*).

[*]Autodesk 3DS Max is a registered trademark of Autodesk Inc.

The *XPairIt* Application Programming Interface (API) offers similar styles of docking methodology and we show how some of these can be combined to improve flexibility and scoring/ranking in peptide-protein docking. *XPairIt* works as a controller code for *PyRosetta, NAMD, PSFGen, STRIDE, VMD, APBS,* and *GAMESS,* and allows a user to move information between these software packages for custom simulations (*1*, *4*, *34–37*). Additionally, the *XPairIt* API and external software packages are tailored for use on high-performance computing (HPC) systems, which is critical for the search of unknown peptide binding locations and the optimization of peptide sequences for improved protein binding.

With existing state-of-the-art in mind, the authors use a bottom-up approach to create a connection between the various styles of simulation software and improve interoperability of the software and their methods. The *XPairIt* API applied to peptide docking joins the external molecular dynamics package *NAMD* and docking software *PyRosetta* to enhance peptide-protein binding, which we refer to as the *XPairIt Docking Protocol* (*4*, *34*). When contrasted with other multi-method docking software, use of the *XPairIt Docking Protocol* offers more detailed control of docking simulations and the ability to run simulations on HPC systems. In general, the methods in particular are similar to the aforementioned work connecting *PyMol,* docking, and molecular dynamics; however, there are several differences in the approach taken to construct the simulations, the style in which we use molecular dynamics, and the analysis of the simulation results. Furthermore, the *XPairIt Docking Protocol* is based on the highly extensible *XPairIt* API, which allows for the simple addition and use of other custom code or external software packages. In the following text, we provide our approach for peptide docking with the *XPairIt Docking Protocol,* and detail the use of external software to address the challenges inherent to peptides: flexibility and scoring. A previous version of this protocol was reported earlier (*38*).

We organize this work by first outlining the *XPairIt* API structure and philosophy, then develop a more robust and improved style of molecular docking, applied to peptide-protein interactions, in *Section II*. *Section III* illustrates the application of the *XPairIt* docking protocol to a common case study. Analysis and Discussion of the protocol are given in *Section IV*. Finally, we provide conclusions and remarks on the future of the software in *Section V*.

2. Methodology

2.1 *XPairIt* Application Programming Interface (API) Philosophy and Structure

XPairIt incorporates a detailed and customizable style of control and analysis during a simulation run, allowing a user to put in place a method whose behavior is dependent on properties computed in real time. Its structure is in line with other software and their inherent partitioning of simulation components. Namely, the *Etomica Simulator* of the Kofke Group at the *University at Buffalo* and the *LAMMPS Molecular Simulator* of Sandia National Laboratories have both had significant influence on the organization of the *XPairIt* API (though all of the code is original) (*5*, *39*). The basic building blocks of a nanoscale simulation—*Boxes*, *Atoms*, and *Vectors*—are represented in the software as Python objects. *Methods, Integrators, Potentials,* and user-defined code perform operations on these objects to move atoms in space, or compute properties of the system based on atomic positions or types. Through Python's object-oriented structure, a user may extend any one of these types to create a custom *Method*, *Integrator*, or *Potential*, or simply use a combination of these in a unique way as a single Python script to control a simulation or analyze its output. Figure 1 diagrams the main objects in *XPairIt* and how they interact.

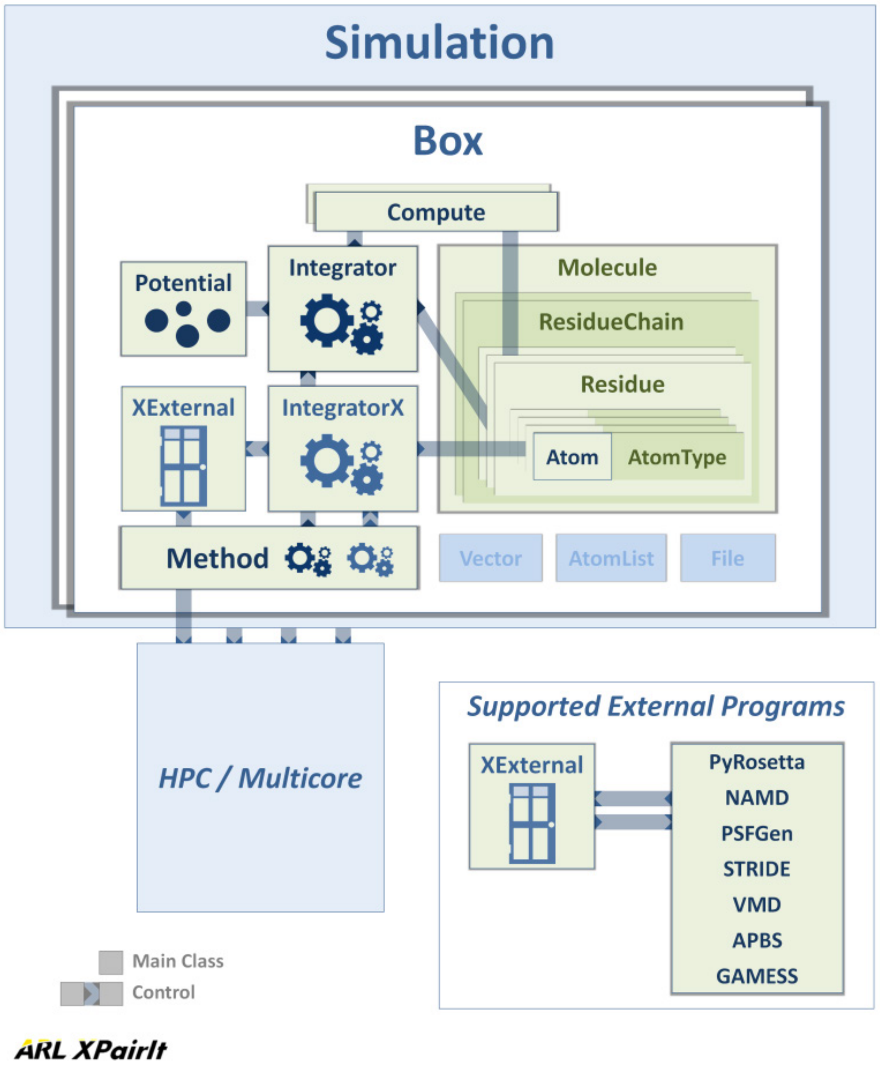


Figure 1. *XPairIt* API Diagram showing simulation components, class hierarchy, control, and connection to external software (XExternal) and hardware (HPC). Supported External Programs: PyRosetta, NAMD, PSFGen, STRIDE, VMD, APBS, and GAMESS (*1*, *4*, *34–37*).

2.2 *XPairIt* API Classes

While not an exhaustive list, the classes below best frame the current version of the *XPairIt API*.

Atom. This object's properties are taken directly from the properties of a real atom; position, velocity, type, and radius. An extension of this object is created for our molecular docking implementation, called *AtomTypeBiological*. This contains other information, such as charge, occupancy, beta parameters, and species name.

Vector. This is a data structure holding a list of *floats*, as well as methods to perform vector operations, such as addition, dot product, cross product, and normalize. *Vector2D*, *Vector3D*, *Vector3DRandom,* and *VectorN* have been created to extend the original *Vector* class. The *Atom* object creates its own *Vector3D* object to track its position.

Box. The simulation volume is defined by this class. *Box* also holds lists of all the atoms that are "inside" the volume, information about boundary conditions and program components such as neighbor lists, and provides methods for creating atoms or changing atom positions.

BoxBiological, an extension of *Box,* is created to handle other properties specific to a biological simulation, such as loading a PDB file.

Simulation. This is the first object that should be created during the development of a simulation with *XPairIt*. It contains structures for the user to connect many high-level simulation components, such as *Integrators* and *Methods*, and allows these components to broadcast information in a one-to-all type message, without knowledge of the other components. Included here is also the software's banner and copyright information for any connected external software.

Compute. These are simple classes that receive a list of *Atoms* and perform operations to calculate specific properties of this list. Examples are *CenterOfMass*, *DihedralAngle*, *SharedInterfaceAtoms*, *RadialDistributionFunction*, *RadiusOfGyration*, *SecondaryStructure*, *HydrogenBonds*, *MeanSquaredDisplacement* and *SurfacePoints*.

Integrator. This class contains methods that perform an operation on the system, typically the *Atoms*. *Integrators* may be created for various molecular dynamics integration schemes, or Monte Carlo sampling styles. These custom integrators can extend the main *Integrator* class to obtain access to callback functions to broadcast commands through *Simulation*.

Method. Typically, this structure contains code for a multi-operation scheme involving a collection of *Integrators*. In the authors' case, several different *Method* classes were created to perform molecular docking simulations. Because of this hierarchy, a user can create a complex simulation with only a few lines of code, using existing *Method* classes.

Residue, ResidueChain, Molecule. These are data structures that map to typical biological groupings of atoms. An amino acid's atoms are in a list within *Residue,* a bonded chain of these amino acids are in a list within *ResidueChain,* and a collection of these bonded chains are in a list within *Molecule*. The methods within these data structures allow for “top-down” and “bottom-up” paths to their parent or child structures. For example, if a user is working with a particular *Atom*, there are methods in place to get to that *Atom’s* parent *Residue, ResidueChain*, and *Molecule*. Conversely, if a user has a *Molecule*, there are methods to choose a particular *ResidueChain* within that *Molecule*, a particular *Residue*, and finally a specific *Atom*. This is useful for passing logical groupings of atoms to *Methods, Integrators*, or *Computes*.

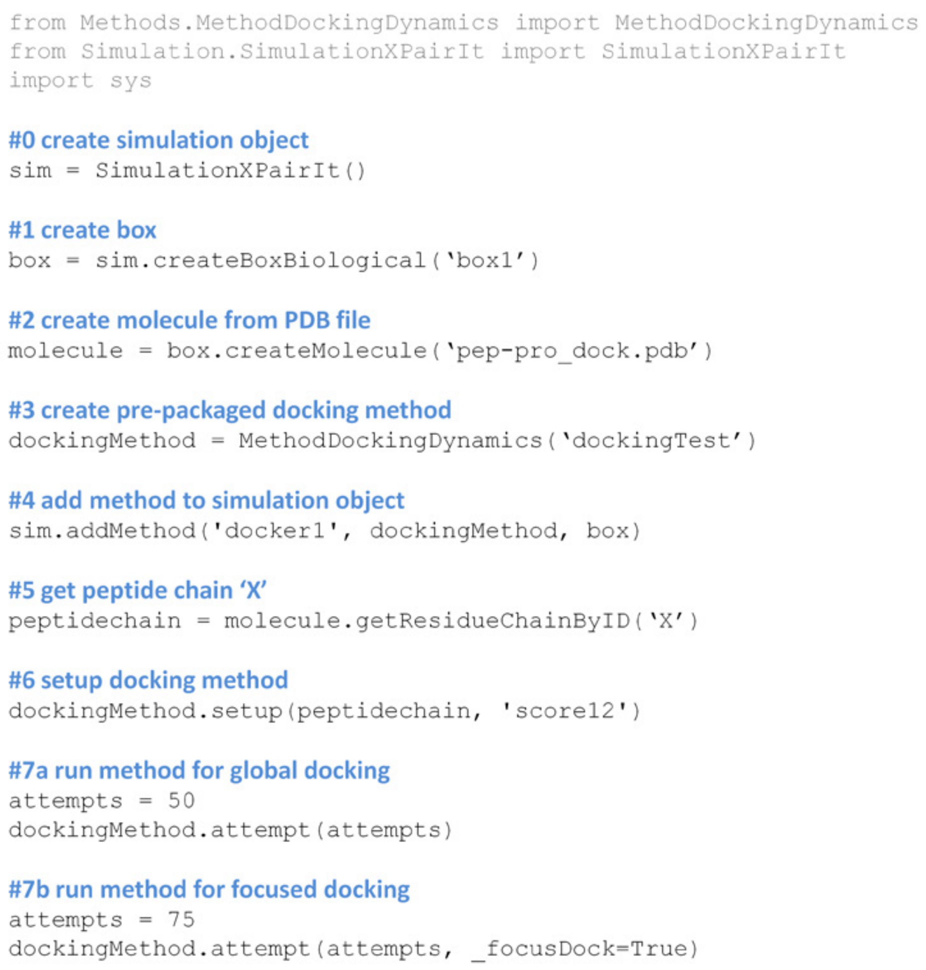
2.3 External Software Interface and Control

*X*____. These are interfaces for the various external software programs used with *XPairIt* and are named using the “X” convention. For example, a nonexhaustive list of current interface classes in *XPairIt* is *XNAMD*, *XPyRosetta*, *XAPBS, XGAMESS*, and *XVMD*. Specialized *Integrators* call these interface classes to allow the user to control an external program.

*IntegratorX*____. A specialized form of the *XPairIt Integrator* which calls a similarly named *X* class. The *IntegratorX* class enables the translation of *XPairIt*-style commands and structure, to and from an external software structure. An example is *IntegratorXNAMD*, where general *Integrator*-style methods call into *XNAMD* to setup, execute, and return output from the NAMD Simulator. These *Integrators* are similar in structure to internal *XPairIt Integrators*, allowing a user to easily perform operations with any type of *IntegratorX*.

2.4 A Simulation With *XPairIt* API

Figure 2 shows a simple Python script using the *XPairIt* API to create a molecular docking simulation, employing several of the simulation building blocks listed in the previous section. First, a *Simulation* object is created to hold a few of the working parts of the simulation. Next, the *Simulation* is used to create a *Box*. Then, a configuration in the form of a PDB file is read in to create a *Molecule, ResidueChain(s), Residue(s)*, and the thousands of *Atom* and *AtomTypeBiological* objects in the simulation. Next, a new docking method is created from existing code. Subsequently, the *Method* is added to the *Simulation*, passing a name, the *Method*, and the *Box* which the *Method* will operate on. Then after variables are created for the peptide chain “X” and number of docking attempts, the setup routine in *Method* is called. Finally, the *Method* is run with chosen parameters.



```
from Methods.MethodDockingDynamics import MethodDockingDynamics
from Simulation.SimulationXPairIt import SimulationXPairIt
import sys

#0 create simulation object
sim = SimulationXPairIt()

#1 create box
box = sim.createBoxBiological(‘box1’)

#2 create molecule from PDB file
molecule = box.createMolecule(‘pep-pro_dock.pdb’)

#3 create pre-packaged docking method
dockingMethod = MethodDockingDynamics(‘dockingTest’)

#4 add method to simulation object
sim.addMethod('docker1', dockingMethod, box)

#5 get peptide chain ‘X’
peptidechain = molecule.getResidueChainByID(‘X’)

#6 setup docking method
dockingMethod.setup(peptidechain, 'score12')

#7a run method for global docking
attempts = 50
dockingMethod.attempt(attempts)

#7b run method for focused docking
attempts = 75
dockingMethod.attempt(attempts, _focusDock=True)
```

Figure 2. *Python* script example of a simple *XPairIt* docking simulation.

2.5 Improved Docking Methods for Peptide Systems with *XPairIt* Docking Protocol

Before we begin our *global docking search*, we first equilibrate the peptide and protein. Simulation of the docking partners, peptide and protein, proceed separately. *XPairIt* drives an NPT simulation at 1.0 atm and 300 K in *NAMD*, using the *CHARMM* interatomic potential with *TIP3P* water (*40*, *41*). The final structure of the protein from equilibration simulations is minimized within *NAMD*. Atomic positions are captured by *XPairIt* and sent to *PyRosetta*, and then the protein's residue sidechain positions are sampled and minimized using the *Rosetta repacking* scheme. This protein structure is then used in all docking runs. For peptide equilibration, starting from a linear structure, the peptide is equilibrated for several nanoseconds. This dynamics trajectory is saved as a series of 1000 snapshots, which are randomly selected as the peptide's starting structure in subsequent docking runs. This is also known as *ensemble docking*.

The docking portion of this protocol is separated into two stages: (1) the initial docking run, testing peptide binding over the entire protein surface, and (2) a more focused docking run, testing probable binding locations on the protein. The first stage is composed of 2000 to 5000 simulations, where the exact number varies with protein size. Each simulation begins by drawing a random peptide structure from the initial dynamics trajectory and placing it in a simulation box with an equilibrated, minimized protein structure. Within *XPairIt*, atom positions are sent to *PyRosetta*. Controlling *PyRosetta*, the partners are randomly rotated around their centers of mass and moved into contact with one another, until any pair of surface atoms of each partner is approximately 4.0Å apart. This creates starting structures where the peptide is placed at a uniform distribution of points on the protein's surface.

For step (i) of the *XPairIt Docking Protocol* stage 1, these simulations (all 2000 to 5000 of them) are run using the *Rosetta DockingMCM* method and the *score12* score function, where the lowest energy structure is determined from sampling small rotation and translation moves of the peptide. *Rosetta's DockingMCM* method also samples sidechain rotamers of the peptide and protein. After step (i) is finished, atom positions are analyzed for structural and energetic properties in a bookkeeping step, and sent to *NAMD* for step (ii). Here, molecular dynamics is performed on the peptide and protein atoms within 15.0Å of the peptide, leaving other atoms fixed. This simulation is performed at 300 K using the *Generalized-Born Implicit Solvent (GBIS+SASA)*, for 2.0 picoseconds (ps), and then the structure is minimized for 3000 conjugate gradient steps (*42*). Finally, for step (iii), the structure's sidechains are again repacked with *Rosetta*, based on the *score12* scoring function, and this structure is exported as a PDB file. A typical change in peptide position for the three main steps of the *XPairIt Docking Protocol* is shown in figure 3. The (i) docking, (ii) dynamics + minimization, (iii) repacking process is repeated 10 times for additional random rotations of the peptide. After the first stage of the docking protocol is complete, 2000 to 5000 simulations are run using a random peptide structure and minimized protein structure, and we produce 20,000 to 50,000 probable bound structures.

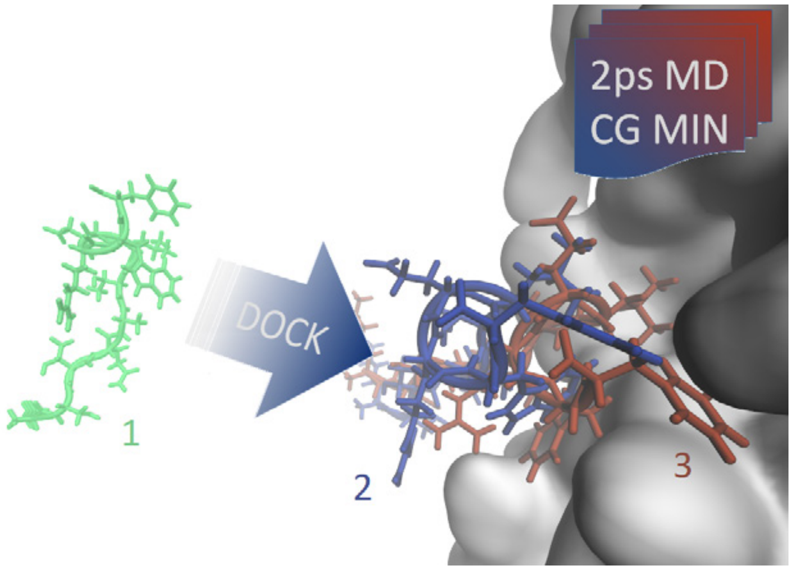


Figure 3. Example positions of peptide during an *XPairIt* docking simulation. (1) Initial placement, shown in *green*. (2) Docked peptide, in *blue*. (3) Minimized structure using conjugate gradient method after 2 ps molecular dynamics run, in *red*.

The second stage in the *XPairIt Docking Protocol* consists of many *focused docking* simulations, which restrict the peptide to sample in only certain areas of the protein. To identify these areas, the top 1000 (of 20,000 to 50,000) docked structures from stage 1 are sorted based on total energy, and then ranked by their interface using *Rosetta's score12* score function. This is shown in *equation 1*. Sorting is performed first based on total energy. This removes any spurious results and restricts our sampling to more likely configurations. A further sort is then performed by interface energy to capture favorable interactions between the peptide and protein.

$$E_{Interface} = E_{Total} - E_{Peptide} - E_{\Pr otein} \qquad (1)$$

These top 25 interface energy structures are then clustered using *XPairIt*, based on the peptide heavy atom distance from each protein heavy atom using a cutoff distance of 4.0Å. This calculation generates a list of protein residues that have contact with the peptide for each structure, and these contacts are counted and totaled for the top 25 structures. For example, a particular residue on the protein may be contacted by the peptide a total number of five times when all 25 structures are analyzed. From these totals, the average number of peptide contacts per protein residue is computed, and those residues that have contact amounts larger than one standard deviation from the mean are recorded as possible binding locations. Next, the structure with the best interface energy is identified at each of these possible binding locations and saved as a starting structure for a stage 2 *focused dock* simulation. An example of this clustering is show in the subsequent section.

Each simulation for the next stage begins by using the new starting structures from clustering, and again drawing a random peptide structure from the equilibration trajectory. This random peptide structure is then moved to the center of mass of the new starting structure. 1000 to 3000 of these simulations are run for each probable binding location and the (i) docking, (ii) dynamics + minimization, and (iii) repacking steps are repeated for 10 different orientations of the peptide. 10,000 to 30,000 docked structures now generated for each possible binding location are ranked by their total energy using *Rosetta's score12* function, and top energies for each location are compared to determine the likely binding location. The final result in the *XPairIt Docking Protocol* is sorted and determined by the total energy.

3. Results and Discussion

We test the *XPairIt Docking Protocol* and present results from a case study, docking a short peptide to a small protein. For this test we choose the 1RXZ system from the *Protein Data Bank*, which has been previously studied with varying degrees of success, using two different docking schemes (*20*, *27*, *44*). This particular case presents added complexity to the simulation of peptide docking in the form of solvent and receptor induced peptide structure and probable receptor flexibility. The two partners here are the 245 residue DNA polymerase sliding clamp protein, *aPCNA*, and an 11-mer peptide (KSTQATLERWF) created from the binding motif of the *Flap EndoNuclease-1* (*aFEN-1)* (*45*). Equilibration with peptide trajectory sampling, one stage of *global docking*, and one stage of *focused docking* are performed according to the previously outlined *XPairIt Docking Protocol*.

3.1 Stage 1 Global Docking

After equilibration of the two partners and creation of the peptide trajectory, we ran 3000 simulations in the *global docking* stage. The resulting docked structures are sorted by total energy, and the top 1000 are then ranked by interface energy. Figure 4 presents an overlay of the top 25 docked structures based on interface energy. Protein residues with six or more contacts, and used in the formulation of stage 2 *focused docking* starting location(s), are shown with a shaded surface.

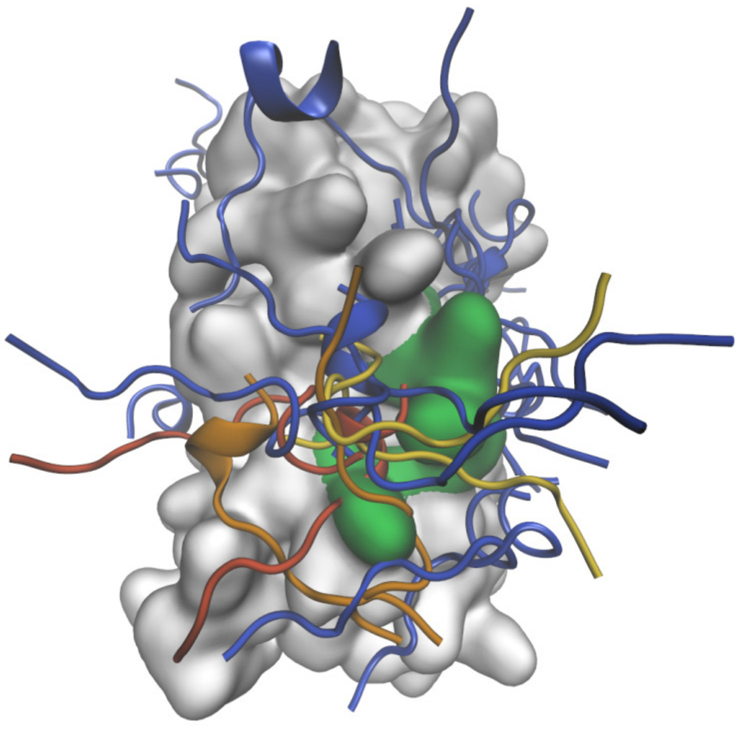


Figure 4. Stage 1 *global docking* results presented as overlay of the top 25 docked structures. Structures are ranked by their interface energy using Rosetta's *score12*. Frequently contacted protein residues are shown with a shaded surface.

In figure 5, the top 25 structures based on interface energy are clustered and plotted on a raw-data histogram, showing specific protein residues (*Residues on A*) in contact with the peptide. For the protein residues with corresponding contact, the average number of peptide contacts for all 25 structures is 2.60 and the standard deviation is 2.47. This creates a contact integer number cutoff of 6 or greater. From the data illustrated in figure 5, we identify protein residues 50, 121–126, 218–220 as having 6 or more contacts when we sample the top 25 structures.

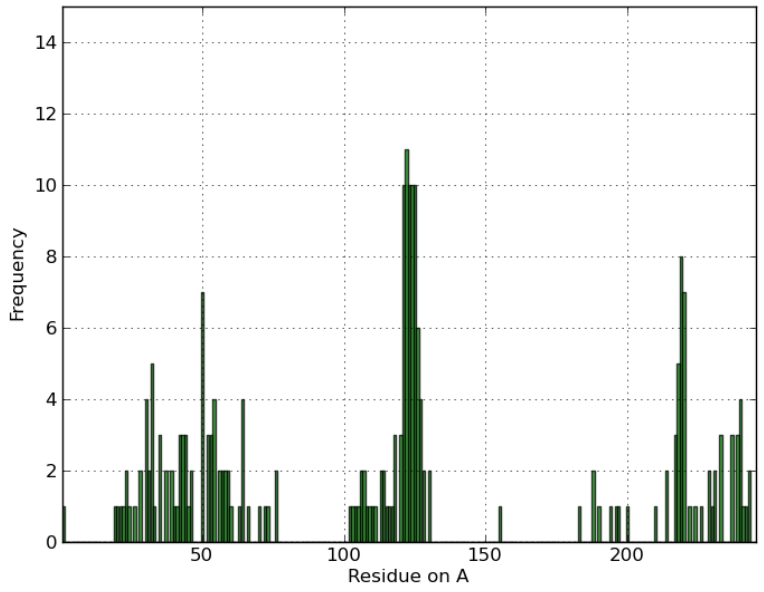


Figure 5. Stage 1 *global docking* results presented as contact histogram. Number of peptide residues (Frequency) within 4.0Å of particular protein residues (Residues on A) for the top 25 docked structures. Structures ranked by their interface energy using Rosetta's *score12*.

3.2 Stage 2 Focused Docking

Before beginning stage 2 *focused docking*, we generated starting configurations by identifying the best ranked structure based on interface energy at each of the locations determined from the clustering of stage 1 *global docking* results. A list of the locations on the protein, with corresponding number of contacts and best ranked structure, are in table 1. From this list, four unique docked structures—*#3*, *#5*, *#7*, and *#8*—were identified for starting structures, shown in figure 6. 1000 docking simulations were then run for each docked starting structure, using peptide configurations randomly drawn from the initial equilibration trajectory and moved to the docked structure peptide's center of mass. For each simulation, 10 rounds of docking were performed using random rotations of the peptide to start each dock. Stage 2 *focused docking* simulations are run as described in the previous section, and results are ranked by total energy using the *Rosetta* score12 score function.

Table 1. Stage 1 *global docking* results clustered as list of protein residues with six or more peptide contacts in the top 25 structures. Structures ranked by interface energy.

Protein Residue	No. of Contacts in Top 25	Best Ranked Structure	Interface Energy (Rosetta Units)
50	7	#3	–13.834
121	10	#3	—
122	11	#3	—
123	10	#7	–13.046
124	10	#5	–13.339
125	11	#5	—
126	7	#5	—
218	6	#8	–12.589
219	9	#5	—
220	7	#8	—

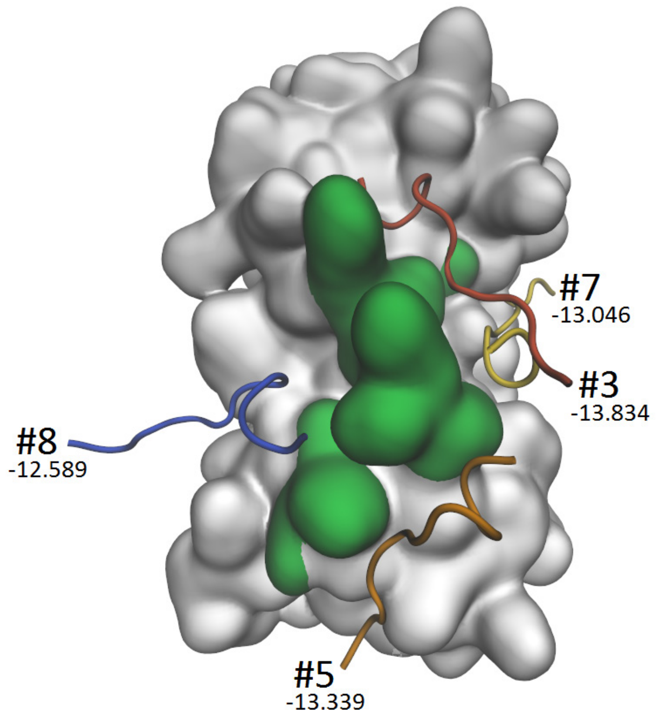


Figure 6. Stage 1 *global docking* top structures based on interface energy for protein residues with sufficient contact. Peptides are labeled with their rank for the top 25 structures based on interface energy. Shown as a shaded surface are protein residues corresponding to histogram peaks in figure 5.

Results for the top ranked structures for each stage 2 *focused docking* starting location are shown in table 2. Here, structures for each of the four previously identified locations are individually ranked by total energy, and the structures with the lowest total energy for each location are compared. In table 2, these top structures are listed arranged by total energy, with additional details about their starting location, interface energy, and RMSD. Validation of these results is conducted by computing the final complex's peptide RMSD from the equilibrated crystal structure of the docked complex. In comparing these values, it is clear that our top docked structure is not within a reasonable distance of the actual binding location. However, investigation of the rank 2 structure's peptide RMSD shows very good agreement with the equilibrated crystal structure. Subsequent analysis of the two remaining structures shows poor agreement with the equilibrated crystal structure. Lacking knowledge of the RMSD, differentiation of the top three structures based on total energy is difficult when one accounts for thermal fluctuations of the peptide-protein complex.

Table 2. Stage 2 *focused docking* results for each starting structure, ranked by total energy.

Rank	**Starting Location**	**Total Energy (Rosetta Units)**	**Interface Energy (Rosetta Units)**	**Peptide RMSD from Crystal Structure (Å)**
1	G.#3	–551.133	–10.712	25.941
2	G.#8	–549.232	–21.541	3.386
3	G.#5	–547.761	–13.798	20.112
4	G.#7	–541.183	–8.739	32.135

In an experimental collaboration, studying a system with an unknown binding location, one might choose to explore additional simulation methods at this phase of the protocol to help differentiate the structures. Methods such as a long molecular dynamics run to insure binding stability or a free energy method to compute a thermodynamically accurate binding energy could be used. Depending on instrument throughput, a search space reduced to a handful of structures may even be welcomed by the experimental team, who could now conduct a manageable number of single or double mutation studies to check binding location. In our case we continue with further analysis and look also at the interface energy for each of the top structures. Of the top three structures, rank 2 possesses nearly double the interface energy of the other two structures with similar total energy. The structure's low RMSD when compared to the crystal structure and low-interface energy make it an ideal candidate for a probable binding location. We continue with analysis of this structure and comment on different aspects of the *XPairIt Docking Protocol*.

3.3 Analysis of Rank 2 Structure

The rank 2 structure from stage 2 *focused docking* is listed in table 2 and its starting structure (G.#8) is shown in figure 7. The rank 2 result shows good agreement with the equilibrated crystal structure, and peptide RMSDs for structures aligned by protein alpha-carbon atoms are 7.595Å for G.#8 and 3.386Å for rank 2. For RMSD calculation, although both structures were aligned to the equilibrated crystal structure using only protein alpha-carbon positions, there are several highly flexible parts of the protein within the peptide binding region. It is likely that for the rank 2 structure the peptide may be bound in a very similar configuration to the equilibrated crystal structure, but due to the flexible regions, RMSD values may be affected by only partial alignment of the protein during the value's computation. RMSD for the protein alpha-carbon components in the *focused* structure vs. the crystal structure is 1.715Å.

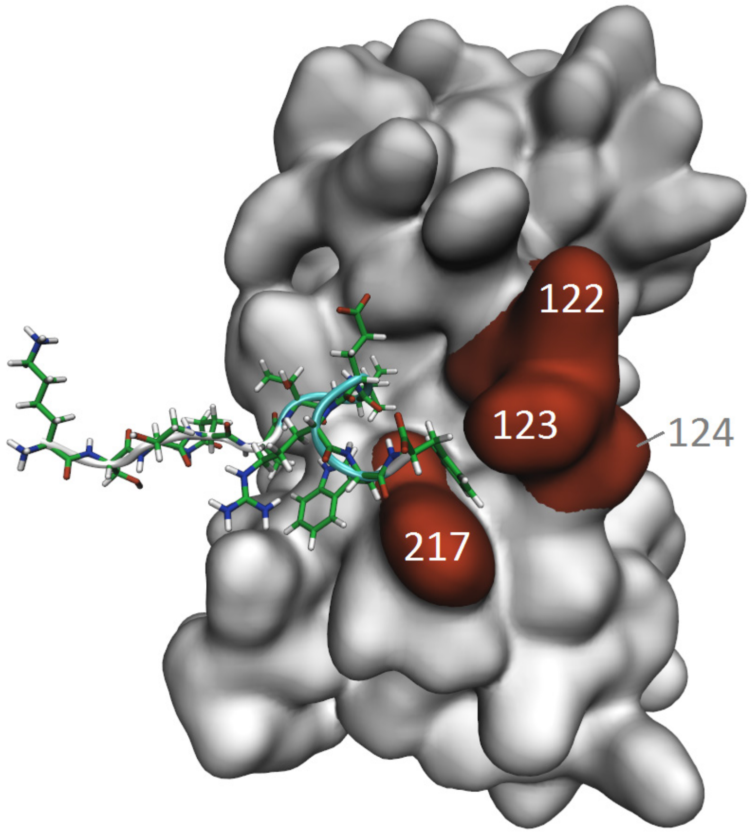


Figure 7. Global dock #8 structure, one of four starting locations for focused docking, with most frequently contacted protein residues highlighted.

Analysis of the *CHARMM* interface energies with *GBIS w/SASA* used in the molecular dynamics and minimization step (ii) of both docking rounds provides further description of the peptide-protein interaction. Shown in table 3, when applied to the final structures of *global* and *focused* rounds of docking, interface energies favor structure #8 (*global*) and #2 (*focused,* from G.#8), correctly identifying the docked structures which best resemble the 1RXZ crystal structure. Additionally, the effects of solvent and pair-wise electrostatic interactions are evident, but do not suggest any aid in differentiating structures. When comparing the overall interface energy from *CHARMM+GBIS w/ SASA* and the *CHARMM van der Waals (vdW)* component, the differences between these two properties for *focused docking* #1 (G.#3) and *focused docking* #2 (G.#8) are similar, meaning the remaining energetic contributions from electrostatics and solvent are similar. Consequently, computation of the *CHARMM* interface energy with *GBIS w/ SASA* correctly identifies the preferred bound structure, indicating that pair-wise electrostatics and solvent driving forces play some part in the interface energy of the docked structures. However, since the contribution from electrostatics and solvent in aggregate are positive, and also similar for both *focused docking* results, it is likely they do not play an active role in forming these particular docked structures. Our results show that the *CHARMM vdW* component is the deciding factor in structure differentiation for 1RXZ.

Table 3. Final interface energy of three bound structures in *score12, CHARMM* with Generalized-Born implicit solvent (*GBIS*), and *CHARMM's vdW* component.

Structure	**Score12 (Rosetta Units)**	**CHARMM+GBIS (kcal/mol)**	**CHARMM vdW Component (kcal/mol)**
Global Dock #3	–13.834	–19.736	–37.183
Global Dock #8	–12.589	–23.315	–40.335
Focused Dock #1 (from G.#3)	–10.712	–13.835	–31.728
Focused Dock #2 (from G.#8)	–21.541	–38.780	–56.318
Crystal Structure	–22.543	–52.246	–70.095

3.4 Effects of Molecular Dynamics Within the *XPairIt* Protocol

Finally, we have an opportunity to once again analyze the effect of molecular dynamics on peptide docking results (*38*). In table 4 we list the *Rosetta score12* interface energies at various steps along both docking stages. There is a marked decrease in interface energy as we employ *NAMD* molecular dynamics and minimization, as well as a final repacking by *Rosetta.* We reiterate that during the step (i) *Rosetta Dock*, the peptide is translated and rotated, and peptide and protein interface sidechains are repacked, similar to step (iii). Backbone angles and bond lengths remain fixed. With the addition of molecular dynamics and minimization in step (ii), the peptide and protein atoms are allowed to relax and move with the effects of temperature and interatomic forces, creating greater contact between the peptide and protein surface, and decreasing the interface energy. We attribute the second decrease in interface energy in step (iii) by *Rosetta sidechain repacking* to a discrepancy between the *CHARMM* and *score12* energy functions. After step (ii) in the *XPairIt Docking Protocol* simulations, atom positions do not change and are sent from *NAMD*, back to *XPairIt*, and then directly to *PyRosetta* for final sidechain repacking. Additionally, a closer look at the effects of dynamics on the docked structure is illustrated in figure 8, where we compare the per residue interface energy of the peptide after step (i) and after step (iii) of stage 1 *global docking*. The *LEFT* plot shows minimal contact of the peptide after step (i) *Rosetta Dock*, when compared to step (iii) *Rosetta sidechain repacking* on the *RIGHT* plot.

Table 4. Interface energies for docked structures at each step of the *XPairIt Docking Protocol*. Energies in *Rosetta* units.

Structure	(i) Rosetta Dock	(ii) NAMD MD + Minimization	(iii) Rosetta Repack Sidechains
Global Dock #8	–1.525	–8.843	–12.589
Focused Dock #1	–10.241	–16.978	–21.541
Equilibrated Crystal Structure	—	—	–22.543

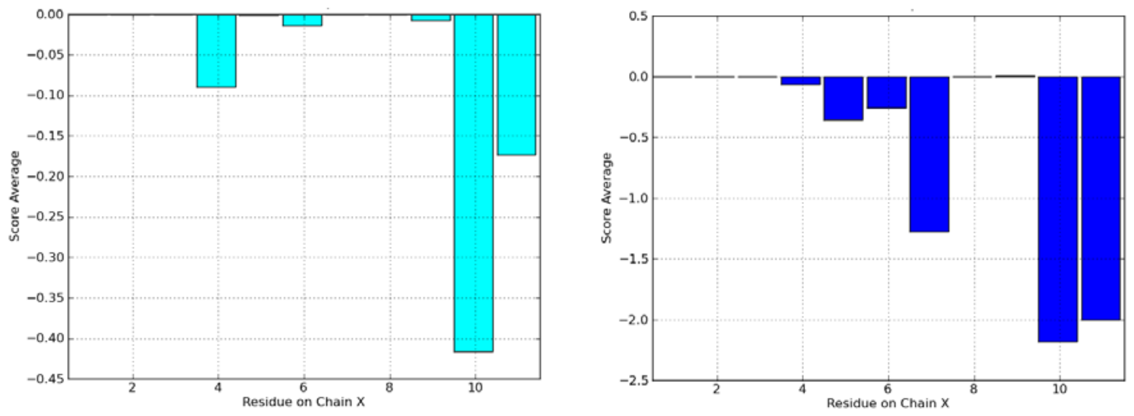


Figure 8. *Rosetta score12* interface energy per peptide residue (Chain X) for stage 1 *global docking* #8 structure: *LEFT* step (i) *Rosetta Dock* and *RIGHT* step (iii) *Rosetta Repack Sidechains*, after *NAMD MD + Minimization*. Peptide sequence: KSTQATLERWF. Note difference in ordinate scales.

4. Summary and Conclusions

The *XPairIt Docking Protocol* is demonstrated here to be a suitable toolkit for the flexible docking of peptides to protein receptors with unknown binding locations. Final docked structures from global docking simulations of the 1RXZ system using the *XPairIt Docking Protocol* show good agreement with the PDB crystal structure. Previous docking studies of this system by Raveh et al. and Dagliyan et al. test the limits of their respective docking protocols (*20–27*). Crystal structure refinement of the 1RXZ system by Raveh et al. using their *FlexPepDock* protocol predicted a final structure of the peptide in helical form, which deviates from the coiled/linear backbone of the peptide in its bound configuration. As reviewed in *Section 1*, using *FlexPepDock*, peptide fragments are sampled to determine optimal structure, and show very good results for all systems, except for 1RXZ and one other reported. We reserve any further comparison to our protocol, as *FlexPepDock* is not used for global peptide docking—this method may be best used with an unknown peptide structure, and a known protein binding pocket. Peptide docking of the 1RXZ structures with discrete molecular dynamics and *MedusaDock* in

Dagliyan et al. shows good agreement with the crystal structure, recovering an RMSD on the same order of magnitude as those reported here. However, Dagliyan et al. suggest that capturing the binding induced protein structural change remains a major challenge. They note that the inclusion of backbone flexibility in the flexible receptor simulations significantly increases the computational time and prefer a fixed backbone with flexible sidechains, where their discrete dynamics regularly run 30–40 ns.

While the *XPairIt Docking Protocol* dynamics runs are on the picosecond timescale and therefore cannot guarantee large scale protein backbone motion, results from 1RXZ illustrate some improvement in this case of receptor flexibility. When compared to the crystal structure, the protein-only alpha-carbon RMSD for our top dock is 1.715Å—a sizable difference, as only protein atoms within 15Å of the peptide move during our simulation and RMSD is averaged over all protein alpha-carbons. Additionally, adding a *focused* round of docking using molecular dynamics allows for significant improvement of protein peptide contact and an improvement of RMSD, shown previously in figure 8 and section 3.4. These results indicate an important change in both peptide and protein backbone structure, and are achieved with a small amount of dynamics simulation time by combining the *Rosetta* and *NAMD* in a coordinated implementation.

Overcoming the current challenges in global peptide docking, such as the representation of flexibility and accurate energetics, require simulation of peptide-protein systems using multiple methods, and accordingly, multiple software packages. The extendable *XPairIt* API provides a unifying code structure, with the ability to successfully implement these software packages and their methods on-the-fly, as customizable docking simulations within the *XPairIt Docking Protocol*.

5. References

1. Humphrey, W.; Dalke, A.; Schulten, K. VMD: Visual Molecular Dynamics. *Journal of molecular graphic* **1996,** *14,* 33–38.
2. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering* **2007,** *9,* 90–95.
3. Elcock, A. H.; Sept, D.; McCammon, J. A. Computer Simulation of Protein−Protein Interactions. *The Journal of Physical Chemistry B* **2001,** *105,* 1504–1518, 2001/03/01.
4. Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. Scalable Molecular Dynamics With NAMD. *Journal of Computational Chemistry* **2005,** *26,* 1781–1802.
5. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **1995,** *117,* 1–19.
6. Berendsen, H. J. C.; van der Spoel, D.; van Drunen, R. GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation. *Computer Physics Communications,* **1995,** *91,* 43–56.
7. Leaver-Fay, A.; Tyka, M.; Lewis, S. M.; Lange, O. F.; Thompson, J.; Jacak, R.; Kaufman, K.; Renfrew, P. D.; Smith, C. A.; Sheffler, W. ROSETTA3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. *Methods Enzymol* **2011,** *487,* 545–574.
8. Halgren, T. A.; Murphy, R. B.; Friesner, R. A.; Beard, H. S.; Frye, L. L.; Pollard, W. T.; Banks, J. L. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening. *Journal of medicinal chemistry* **2004,** *47,* 1750–1759.
9. Molecular Operating Environment (MOE), ed. *H3A 2R7: Chemical Computing Group Inc*., Montreal, QC, Canada, 2012.
10. Hinsen, K. The Molecular Modeling Toolkit: a New Approach to Molecular Simulations. *Journal of Computational Chemistry,* vol. 21, pp. 79-85, 2000.
11. Eastman, P.; Friedrichs, M. S.; Chodera, J. D.; Radmer, R. J.; Bruns, C. M.; Ku, J. P.; Beauchamp, K. A.; Lane, T. J.; Wang, L.-P.; Shukla, D. OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High-Performance Molecular Simulation. *Journal of chemical theory and computation* **2012,** *9,* 461–469.

12. Janin, J.; Henrick, K.; Moult, J.; Eyck, L. T.; Sternberg, M. J. E.; Vajda, S.; Vakser, I.; Wodak, S. J. CAPRI: A Critical Assessment of Predicted Interactions. *Proteins: Structure, Function, and Bioinformatics,* **2003,** *52,* 2–9.

13. Fleishman, S. J.; Whitehead, T. A.; Ekiert, D. C.; Dreyfus, C.; Corn, J. E.; Strauch, E.-M.; Wilson, I. A.; Baker, D. Computational Design of Proteins Targeting the Conserved Stem Region of Influenza Hemagglutinin. *Science* **13 May 2011,** *332,* 816–821.

14. Kaufmann, K. W.; Lemmon, G. H.; DeLuca, S. L.; Sheehan, J. H.; Meiler, J.; Practically Useful: What the Rosetta Protein Modeling Suite can do for you. *Biochemistry* **2010,** *49,* 2987–2998.

15. Gray, J. J.; Moughon, S.; Wang, C.; Schueler-Furman, O.; Kuhlman, B.; Rohl, C. A.; Baker, D. Protein–Protein Docking With Simultaneous Optimization of Rigid-Body Displacement and Side-Chain Conformations. *Journal of Molecular Biology* **2003,** *331*, 281–300.

16. Sivasubramanian, A.; Maynard, J. A.; Gray, J. J. Modeling the structure of mAb 14B7 Bound to the Anthrax Protective Antigen. *Proteins: Structure, Function, and Bioinformatics* **2007,** *70,* 218–230.

17. Chaudhury, S.; Gray, J. J. Identification of Structural Mechanisms of HIV-1 Protease Specificity Using Computational Peptide Docking: Implications for Drug Resistance. *Structure* **2009,** *17,* 1636–1648.

18. Sammond, D. W.; Bosch, D. E.; Butterfoss, G. L.; Purbeck, C.; Machius, M.; Siderovski, D. P.; Kuhlman, B. Computational Design of the Sequence and Structure of a Protein-Binding Peptide. *Journal of the American Chemical Society* **2011,** *133,* 4190–4192.

19. Raveh, B.; London, N.; Schueler-Furman O. Sub-Angstrom Modeling of Complexes Between Flexible Peptides and Globular Proteins. *Proteins: Structure, Function, and Bioinformatics* **2010,** *78,* 2029–2040.

20. Raveh, B.; London, N.; Zimmerman, L.; Schueler-Furman, O. Rosetta FlexPepDock ab-initio: Simultaneous Folding, Docking and Refinement of Peptides Onto Their Receptors. *PLoS One* **2011,** *6,* e18934.

21. Alonso, H.; Bliznyuk, A. A.; Gready, J. E. Combining Docking and Molecular Dynamic Simulations in Drug Design. *Medicinal Research Reviews* **2006,** *26,* 531–568.

22. Audie, J.; Swanson, J. Recent Work in the Development and Application of Protein–Peptide Docking. *Future* **2012,** *4,* 1619–1644.

23. Lin, J.-H.; Perryman, A. L.; Schames, J. R.; McCammon, J. A. Computational Drug Design Accommodating Receptor Flexibility: the Relaxed Complex Scheme. *Journal of the American Chemical Society* **2002,** *124,* 5632–5633.

24. Okimoto, N.; Futatsugi, N.; Fuji, H.; Suenaga, A.; Morimoto, G.; Yanai, R.; Ohno, Y.; Narumi, T.; Taiji, M. High-Performance Drug Discovery: Computational Screening by Combining Docking and Molecular Dynamics Simulations. *PLoS Computational Biology* **2009,** *5,* e1000528.

25. Antes, I. DynaDock: A New Molecular Dynamics-Based Algorithm for Protein–Peptide Docking Including Receptor Flexibility. *Proteins: Structure, Function, and Bioinformatics* **2010,** *78,* 1084–1104.

26. Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. AutoDock4 and AutoDockTools4: Automated Docking With Selective Receptor Flexibility. *Journal of Computational Chemistry* **2009,** *30,* 2785–2791.

27. Dagliyan, O.; Proctor, E. A.; D'Auria, K. M.; Ding, F.; Dokholyan, N. V. Structural and Dynamic Determinants of Protein-Peptide Recognition. *Structure* **2011,** *19,* 1837–1845.

28. Nowosielski, M.; Hoffmann, M.; Kuron, A.; Korycka-Machala, M.; Dziadek, J. The MM2QM tool for Combining Docking, Molecular Dynamics, Molecular Mechanics, and Quantum Mechanics. *Journal of computational chemistry* **2012.**

29. Whalen, K. L.; Chang, K. M.; Spies, M. A. Hybrid Steered Molecular Dynamics-Docking: An Efficient Solution to the Problem of Ranking Inhibitor Affinities Against a Flexible Drug Target. *Molecular informatics* **2011,** *30,* 459–471.

30. WDeLano, . L. *The PyMOL Molecular Graphics System*, 2002.

31. Lill, M. A.; Danielson, M. L. Computer-Aided Drug Design Platform Using PyMOL. *Journal of computer-aided molecular design* **2011,** *25,* 13–19.

32. Seeliger, D.; De Groot, B. L. Ligand Docking and Binding Site Analysis With PyMOL and Autodock/Vina. *Journal of Computer-Aided Molecular Design* **2010,** *24,* 417–422.

33. Johnson, G. T.; Autin, L.; Goodsell, D. S.; Sanner, M. F.; Olson, A. J. < i> ePMV</i> Embeds Molecular Modeling into Professional Animation Software Environments. *Structure* **2011,** *19,* 293–303.

34. Chaudhury, S.; Lyskov, S.; Gray, J. J. PyRosetta: A Script-Based Interface for Implementing Molecular Modeling Algorithms Using Rosetta. *Bioinformatics* **2010,** *26,* 689–691.

35. Frishman, D.; Argos, P. Knowledge-Based Protein Secondary Structure Assignment. *Proteins: Structure, Function, and Bioinformatics* **2004,** *23,* 566–579.

36. Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. Electrostatics of Nanosystems: Application to Microtubules and the Ribosome. *Proceedings of the National Academy of Sciences* **2001,** *98,* 10037–10041.

37. Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S. General Atomic and Molecular Electronic Structure System. *Journal of Computational Chemistry* **2004,** *14,* 1347–1363.

38. Sellers, M.; Hurley, M. M. In Silico Design of Smart Binders to Anthrax PA. In *SPIE Defense, Security, and Sensing,* 2012; pp 835807-835807-9.

39. Bucher, H. F.; Schultz, A. J.; Kofke, D. A. An Eclipse-Based Environment for Molecular Simulation. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange,* 2005; pp 130–134.

40. MacKerell, Jr, A. D.; Bashford, D.; Bellott, M.; Dunbrack, Jr, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J. a.; Guo, H.; Ha, S. a. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *The Journal of Physical Chemistry B* **1998,** *102,* 3586–3616.

41. Still, W. C.; Tempczyk, A.; Hawley, R. C.; Hendrickson, T. Semianalytical Treatment of Solvation for Molecular Mechanics and Dynamics. *Journal of the American Chemical Society* **1990,** *112,* 6127–6129.

42. Tanner, D. E.; Chan, K.-Y.; Phillips, J. C.; Schulten, K. Parallel Generalized Born Implicit Solvent Calculations With NAMD. *Journal of Chemical Theory and Computation* **2011,** *7,* 3635–3642.

43. London, N.; Movshovitz-Attias, D.; Schueler-Furman, O. The Structural Basis of Peptide-Protein Binding Strategies. *Structure* **2 February 2010,** *18,* 188–199.

44. Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Research* **2000,** *28,* 235–242.

45. Chapados, B. R.; Hosfield, D. J.; Han, S.; Qiu, J.; Yelent, B.; Shen, B.; Tainer, J. A. Structural Basis for FEN-1 Substrate Specificity and PCNA-Mediated Activation in DNA Replication and Repair. *Cell* **2004,** *116,* 39–50.

List of Symbols, Abbreviations, and Acronyms

aPCNA	Proliferating Cell Nuclear Antigen
aFEN-1	Flap EndoNuclease-1
API	application programming interface
CDK2	Cyclin-dependent Kinase-2
CHARMM	Chemistry at Harvard Molecular Mechanics
DNA	Deoxyribonucleic acid
GOLD	Genetic Optimization for Ligand Docking
GPU	graphics processing unit
MC	Monte Carlo
MCM	Monte Carlo Mover
MM/PB-SA	Molecular Mechanics / Poisson-Boltzmann Surface Area
MMTK	Molecular Modeling Toolkit
MOE	Molecular Operating Environment
OPMD	optimized potential molecular dynamics
SMD	steered-molecular dynamics
ePMV	embedded Python Molecular Viewer
HPC	high-performance computing
NAMD	Nanoscale Molecular Dynamics
GBIS	Generalized-Born Implicit Solvent
SASA	solvent accessible surface area
PDB	protein data bank
ps	picoseconds
RMSD	root mean squared displacement

vdW	van der Waals
VMD	Visual Molecular Dynamics

1	DEFENSE TECHNICAL
(PDF)	INFORMATION CTR
	DTIC OCA
2	DIRECTOR
(PDF)	US ARMY RESEARCH LAB
	RDRL CIO LL
	IMAL HRA MAIL & RECORDS MGMT
1	GOVT PRINTG OFC
(PDF)	A MALHOTRA
1	DIR USARL
(PDF)	RDRL WML B
	M SELLERS

INTENTIONALLY LEFT BLANK.